Quartus II Introduction Using Schematic Designs

1 Introduction

This tutorial presents an introduction to the Quartus® II CAD system. It gives a general overview of a typical CAD flow for designing circuits that are implemented by using FPGA devices, and shows how this flow is realized in the Quartus II software. The design process is illustrated by giving step-by-step instructions for using the Quartus II software to implement a very simple circuit in an Altera FPGA device.

The Quartus II system includes full support for all of the popular methods of entering a description of the desired circuit into a CAD system. This tutorial makes use of the schematic design entry method, in which the user draws a graphical diagram of the circuit. Two other versions of this tutorial are also available, which use the Verilog and VHDL hardware description languages, respectively.

The last step in the design process involves configuring the designed circuit in an actual FPGA device. To show how this is done, it is assumed that the user has access to the Altera DE-series Development and Education board connected to a computer that has Quartus II software installed. A reader who does not have access to the DE-series board will still find the tutorial useful to learn how the FPGA programming and configuration task is performed.

The screen captures in the tutorial were obtained using the Quartus II version 11.0; if other versions of the software are used, some of the images may be slightly different.

Contents:

- Typical CAD Flow
- Getting Started
- Starting a New Project
- Schematic Design Entry
- Compiling the Design
- Pin Assignment
- Simulating the Designed Circuit
- Programming and Configuring the FPGA Device
- Testing the Designed Circuit

2 Background

Computer Aided Design (CAD) software makes it easy to implement a desired logic circuit by using a programmable logic device, such as a field-programmable gate array (FPGA) chip. A typical FPGA CAD flow is illustrated in Figure 1.

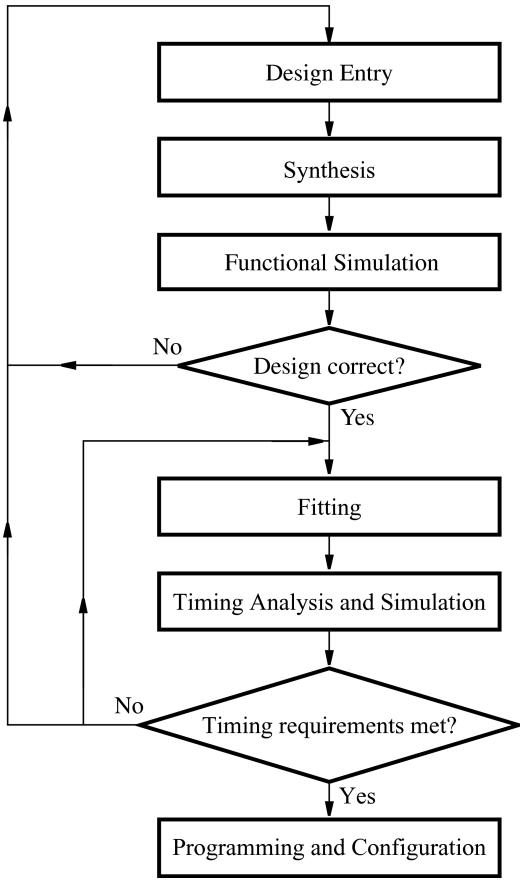


Figure 1. Typical CAD flow.

The CAD flow involves the following steps:

- **Design Entry** – the desired circuit is specified either by means of a schematic diagram, or by using a hardware description language, such as Verilog or VHDL
- **Synthesis** – the entered design is synthesized into a circuit that consists of the logic elements (LEs) provided in the FPGA chip
- **Functional Simulation** – the synthesized circuit is tested to verify its functional correctness; this simulation does not take into account any timing issues

- **Fitting** – the CAD Fitter tool determines the placement of the LEs defined in the netlist into the LEs in an actual FPGA chip; it also chooses routing wires in the chip to make the required connections between specific LEs
- **Timing Analysis** – propagation delays along the various paths in the fitted circuit are analyzed to provide an indication of the expected performance of the circuit
- **Timing Simulation** – the fitted circuit is tested to verify both its functional correctness and timing
- **Programming and Configuration** – the designed circuit is implemented in a physical FPGA chip by programming the configuration switches that configure the LEs and establish the required wiring connections

This tutorial introduces the basic features of the Quartus II software. It shows how the software can be used to design and implement a circuit specified by means of a schematic diagram. It makes use of the graphical user interface to invoke the Quartus II commands. Doing this tutorial, the reader will learn about:

- Creating a project
- Entering a schematic diagram
- Synthesizing a circuit from the schematic diagram
- Fitting a synthesized circuit into an Altera FPGA
- Assigning the circuit inputs and outputs to specific pins on the FPGA
- Simulating the designed circuit
- Programming and configuring the FPGA chip on Altera's DE-series board

3 Getting Started

Each logic circuit, or subcircuit, being designed with Quartus II software is called a *project*. The software works on one project at a time and keeps all information for that project in a single directory (folder) in the file system. To begin a new logic circuit design, the first step is to create a directory to hold its files. To hold the design files for this tutorial, we will use a directory *introtutorial*. The running example for this tutorial is a simple circuit for two-way light control.

Start the Quartus II software. You should see a display similar to the one in Figure 2. This display consists of several windows that provide access to all the features of Quartus II software, which the user selects with the computer mouse. Most of the commands provided by Quartus II software can be accessed by using a set of menus that are located below the title bar. For example, in Figure 2 clicking the left mouse button on the menu named File opens the menu shown in Figure 3. Clicking the left mouse button on the entry Exit exits from Quartus II software. In general, whenever the mouse is used to select something, the *left* button is used. Hence we will not normally specify which button to press. In the few cases when it is necessary to use the *right* mouse button, it will be specified explicitly.

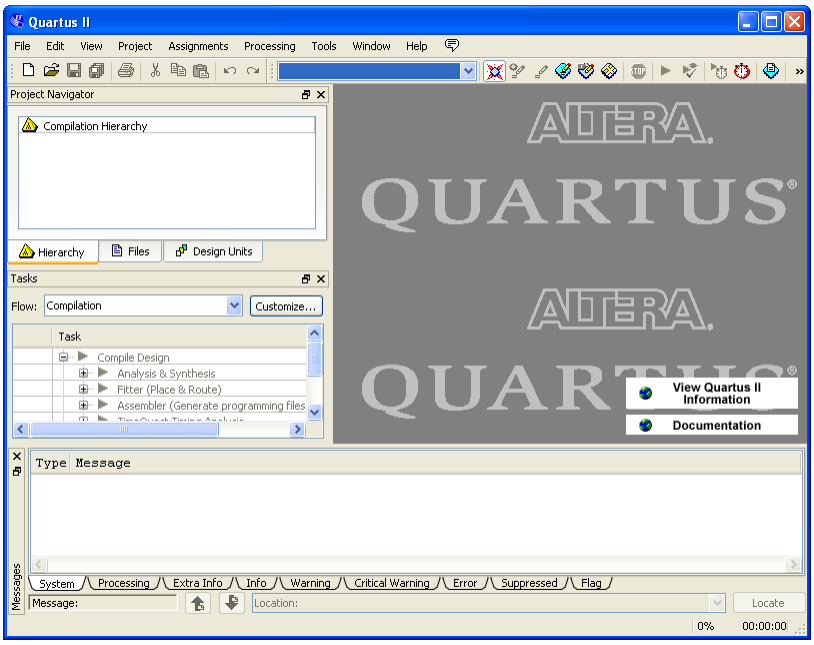


Figure 2. The main Quartus II display.

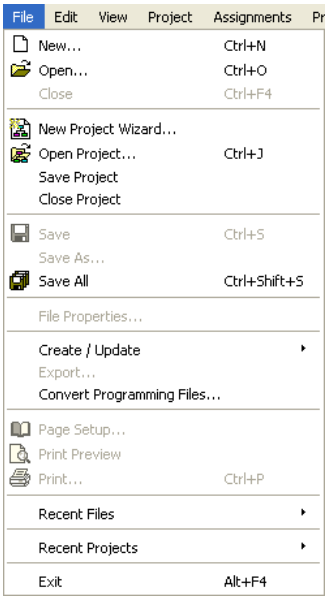


Figure 3. An example of the File menu.

For some commands it is necessary to access two or more menus in sequence. We use the convention Menu1 > Menu2 > Item to indicate that to select the desired command the user should first click the left mouse button on Menu1, then within this menu click on Menu2, and then within Menu2 click on Item. For example, File > Exit uses the mouse to exit from the system. Many commands can be invoked by clicking on an icon displayed in one of the toolbars. To see the command associated with an icon, position the mouse over the icon and a tooltip will appear that displays the command name.

3.1 Quartus II Online Help

Quartus II software provides comprehensive online documentation that answers many of the questions that may arise when using the software. The documentation is accessed from the Help menu. To get some idea of the extent of documentation provided, it is worthwhile for the reader to browse through the Help menu.

If no web browser is specified, Quartus will complain with an error message. To specify a web browser, go to Tools > Options... > General > Internet Connectivity. Specify a path to a web browser in the web browser field.

The user can quickly search through the Help topics by selecting Help > Search, which opens a dialog box into which keywords can be entered. Another method, context-sensitive help, is provided for quickly finding documentation for specific topics. While using most applications, pressing the F1 function key on the keyboard opens a Help display that shows the commands available for the application.

4 Starting a New Project

To start working on a new design we first have to define a new *design project*. Quartus II software makes the designer's task easy by providing support in the form of a *wizard*. Create a new project as follows:

1. Select File > New Project Wizard to reach the window in Figure 4, which asks for the name and directory of the project.
2. Set the working directory to be *introtutorial*; of course, you can use some other directory name of your choice if you prefer. The project must have a name, which is usually the same as the top-level design entity that will be included in the project. Choose *light* as the name for both the project and the top-level entity, as shown in Figure 4. Press Next. Since we have not yet created the directory *introtutorial*, Quartus II software displays the pop-up box in Figure 5 asking if it should create the desired directory. Click Yes, which leads to the window in Figure 6.

Note for IS1500: The New Project Wizard will initially suggest your home folder as a working directory.
At the end of the suggested path, add the following: `/qp/introtutorial`
(We suggest "qp" as the name of the folder that will contain all your Quartus Projects.)
The complete path should now be something like: /home/student/qp/introtutorial
If you use a computer managed by your school, /home/student will be replaced by something completely different.

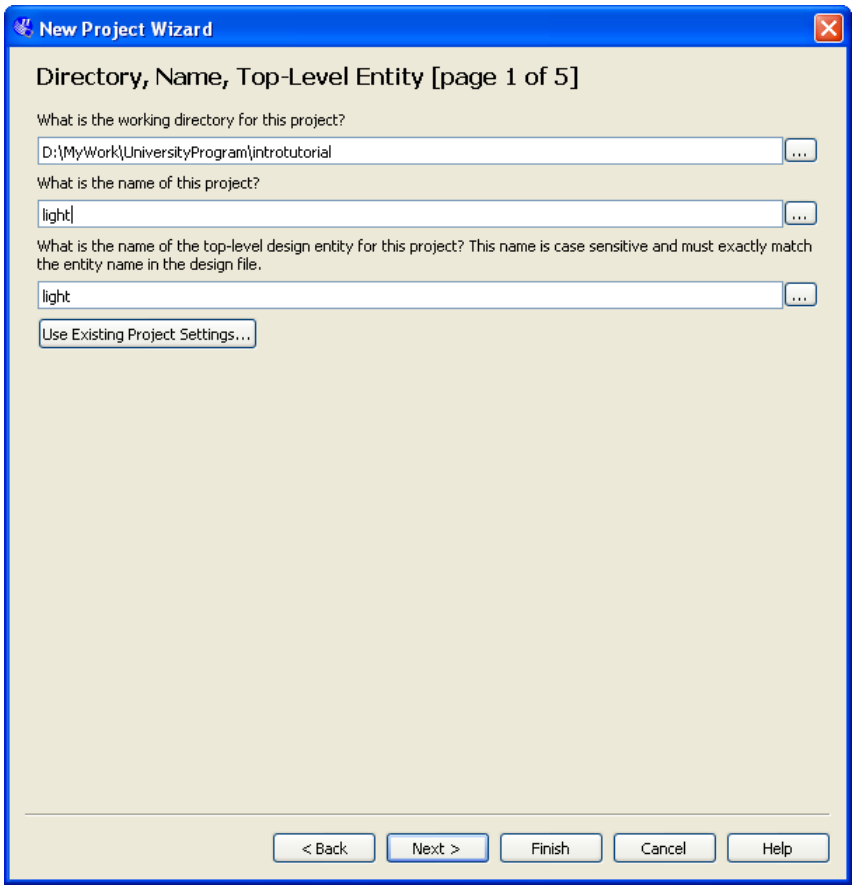

New Project Wizard
Directory, Name, Top-Level Entity [page 1 of 5]
What is the working directory for this project?
D:\MyWork\UniversityProgram\introtutorial
What is the name of this project?
light
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
light
Use Existing Project Settings...
< Back
Next >
Finish
Cancel
Help

Figure 4. Creation of a new project.

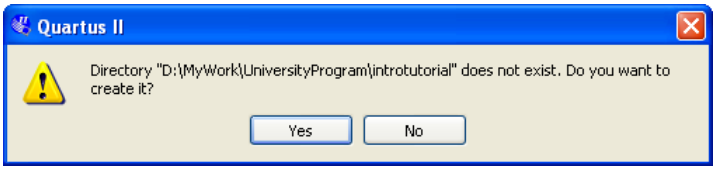


Figure 5. Quartus II software can create a new directory for the project.

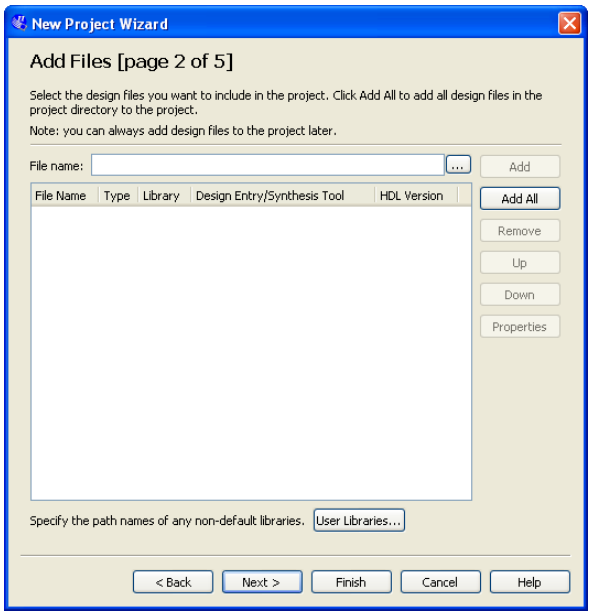


Figure 6. The wizard can include user-specified design files.

3. The wizard makes it easy to specify which existing files (if any) should be included in the project. Assuming that we do not have any existing files, click Next, which leads to the window in Figure 7.

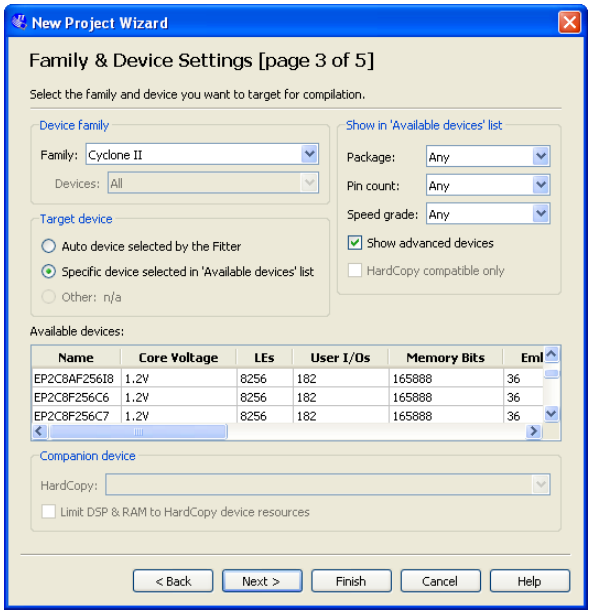


Figure 7. Choose the device family and a specific device.

Note for IS1500: We use the DE2 board, with a Cyclone II EP2C35F672C6 device.
Select Pin count = 672 and Speed = 6 to reduce the number of choices.

4. We have to specify the type of device in which the designed circuit will be implemented. Choose the Cyclone-series device family for your DE-series board. We can let Quartus II software select a specific device in the family, or we can choose the device explicitly. We will take the latter approach. From the list of available devices, choose the appropriate device name for your DE-series board. A list of devices names on DE-series boards can be found in Table 1. Press Next, which opens the window in Figure 8.

Board	Device Name
DE0	Cyclone III EP3C16F484C6
DE1	Cyclone II EP2C20F484C7
DE2	Cyclone II EP2C35F672C6
DE2-70	Cyclone II EP2C70F896C6
DE2-115	Cyclone IVE EP4CE115F29C7

Table 1. DE-series FPGA device names

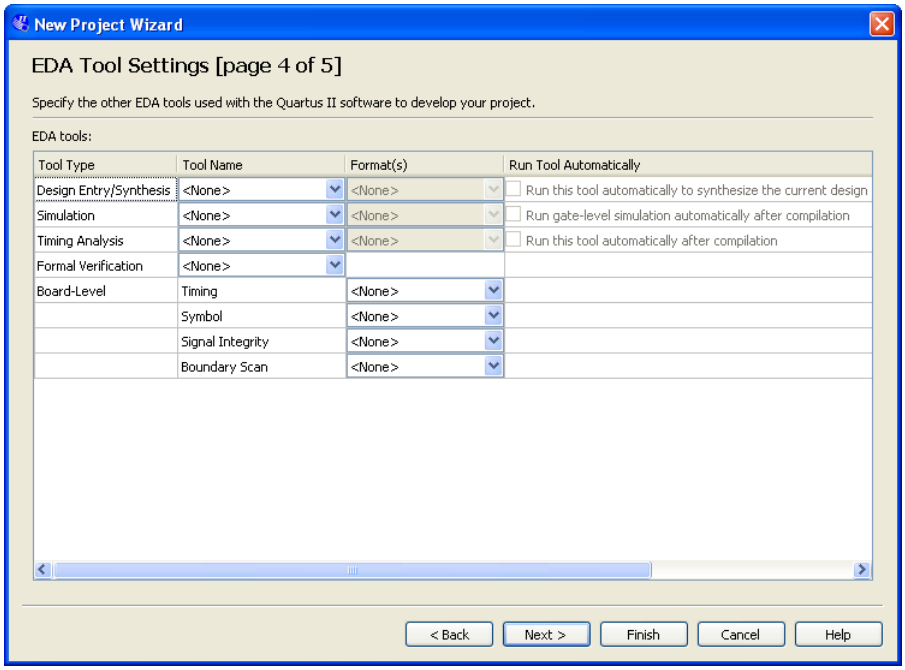


Figure 8. Other EDA tools can be specified.

5. The user can specify any third-party tools that should be used. A commonly used term for CAD software for electronic circuits is *EDA tools*, where the acronym stands for Electronic Design Automation. This term is used in Quartus II messages that refer to third-party tools, which are the tools developed and marketed by companies other than Altera. Since we will rely solely on Quartus II tools, we will not choose any other tools. Press Next.

6. A summary of the chosen settings appears in the screen shown in Figure 9. Press Finish, which returns to the

main Quartus II window, but with *light* specified as the new project, in the display title bar, as indicated in Figure 10.

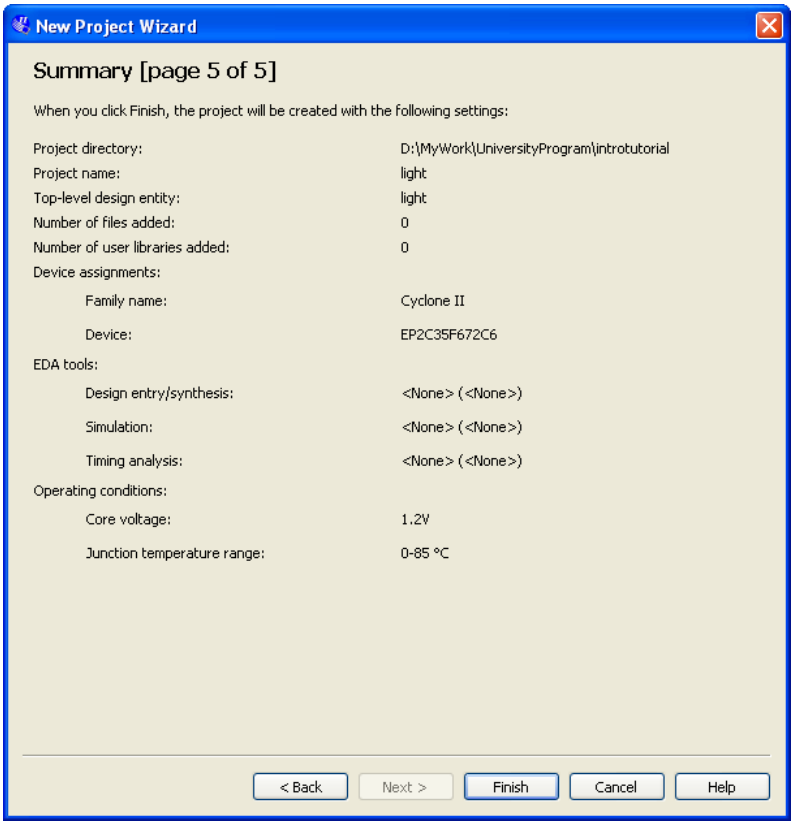


Figure 9. Summary of project settings.

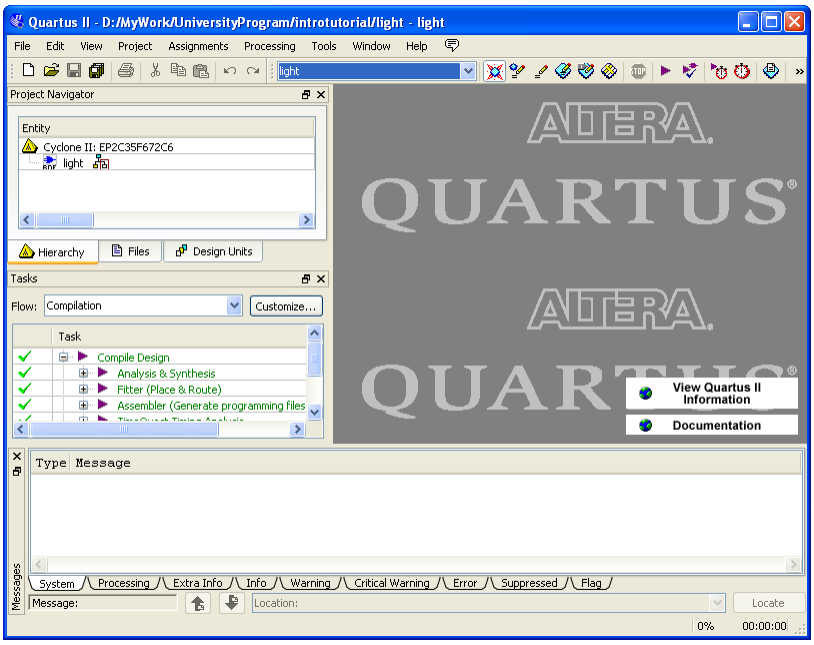
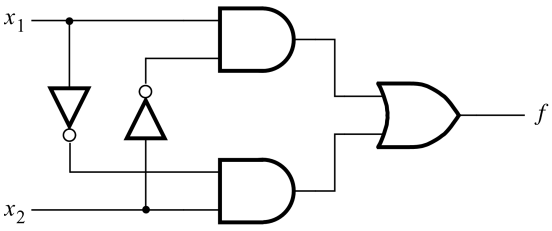


Figure 10. The Quartus II display for a created project.

5 Design Entry Using the Graphic Editor

As a design example, we will use the two-way light controller circuit shown in Figure 11. The circuit can be used to control a single light from either of the two switches, x_1 and x_2, where a closed switch corresponds to the logic value 1. The truth table for the circuit is also given in the figure. Note that this is just the Exclusive-OR function of the inputs x_1 and x_2, but we will implement it using the gates shown.



x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

Figure 11. The light controller circuit.

The Quartus II Graphic Editor can be used to specify a circuit in the form of a block diagram. Select File > New to get the window in Figure 12, choose Block Diagram/Schematic File, and click OK. This opens the Graphic Editor window. The first step is to specify a name for the file that will be created. Select File > Save As to open the pop-up box depicted in Figure 13. In the box labeled Save as type choose Block Diagram/Schematic File (*.bdf). In the box labeled File name type *light*, to match the name given in Figure 4, which was specified when the project was

created. Put a checkmark in the box Add file to current project. Click Save, which puts the file into the directory *introtutorial* and leads to the Graphic Editor window displayed in Figure 14.

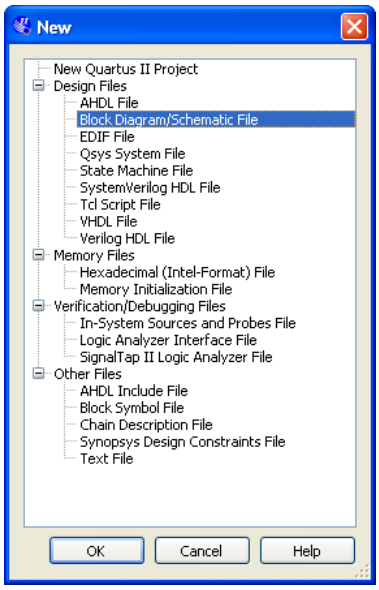


Figure 12. Choose to prepare a block diagram.

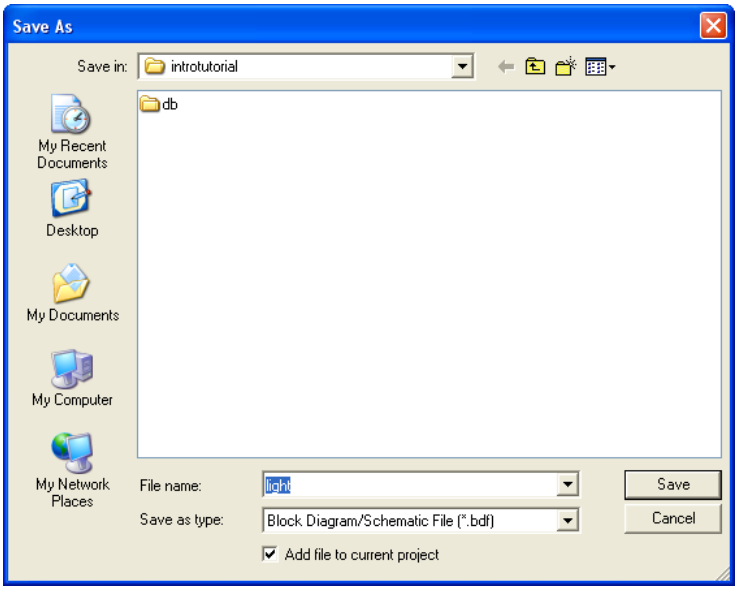


Figure 13. Name the file.

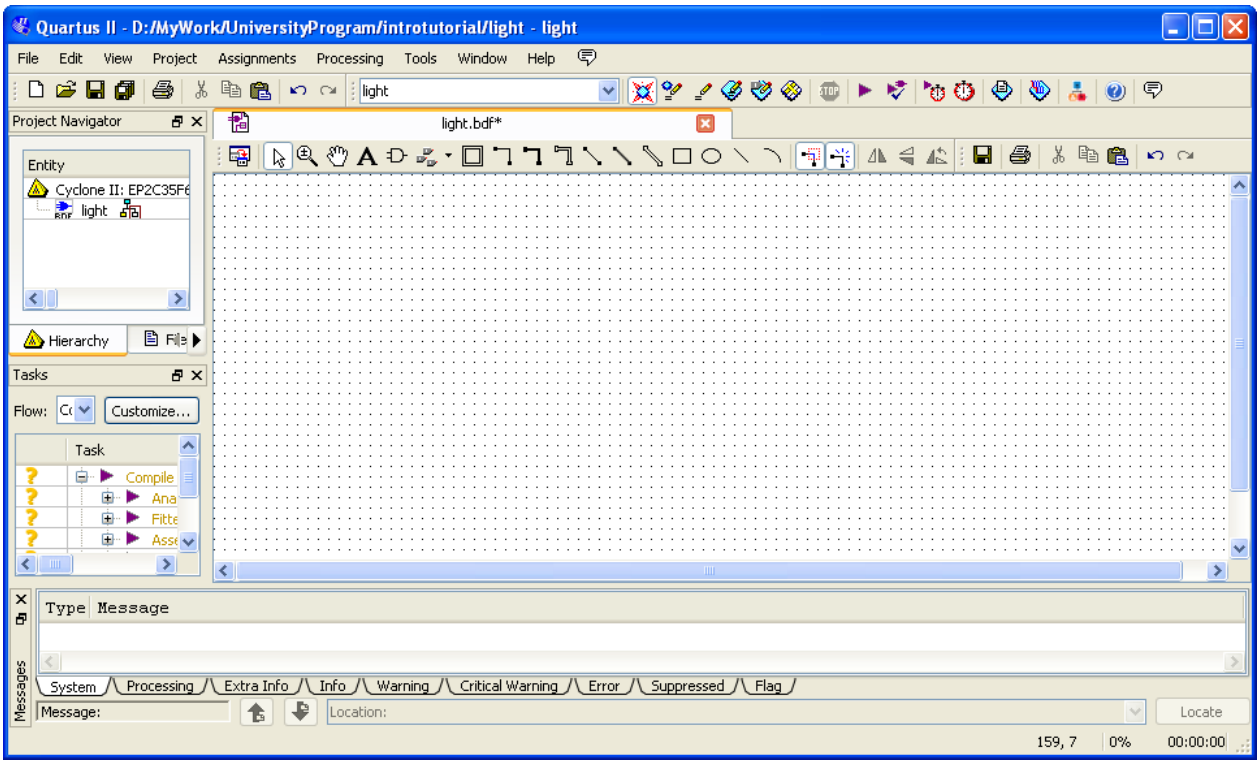


Figure 14. Graphic Editor window.

5.1 Importing Logic-Gate Symbols

The Graphic Editor provides a number of libraries which include circuit elements that can be imported into a schematic. Double-click on the blank space in the Graphic Editor window, or click on the icon in the toolbar that looks like an AND gate. A pop-up box in Figure 15 will appear. Expand the hierarchy in the Libraries box as shown in the figure. First expand *libraries*, then expand the library *primitives*, followed by expanding the library *logic* which comprises the logic gates. Select *and2*, which is a two-input AND gate, and click OK. Now, the AND gate symbol will appear in the Graphic Editor window. Using the mouse, move the symbol to a desirable location and click to place it there. Import the second AND gate by simply moving the mouse pointer to a new position and clicking to place another AND gate symbol there. A symbol in the Graphic Editor window can be moved by clicking the icon in the toolbar that looks like a mouse cursor, then clicking the symbol you want to move and dragging it to a new location with the mouse button pressed. Next, select *or2* from the library and import the OR gate into the diagram. Then, select *not* and import two instances of the NOT gate. Rotate the NOT gates into proper position by using the “Rotate left 90" icon . Arrange the gates as shown in Figure 16.

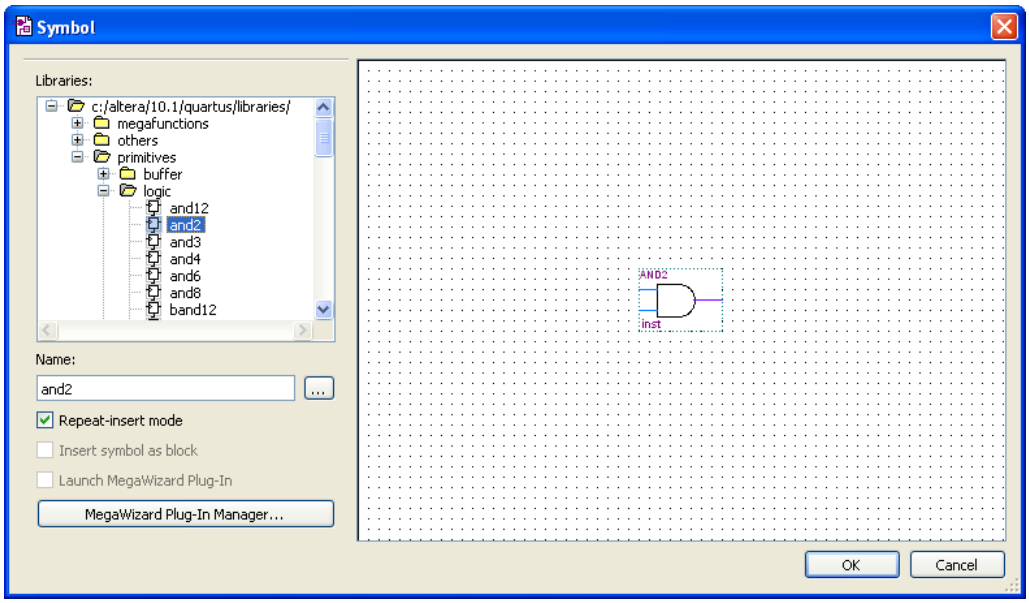


Figure 15. Choose a symbol from the library.

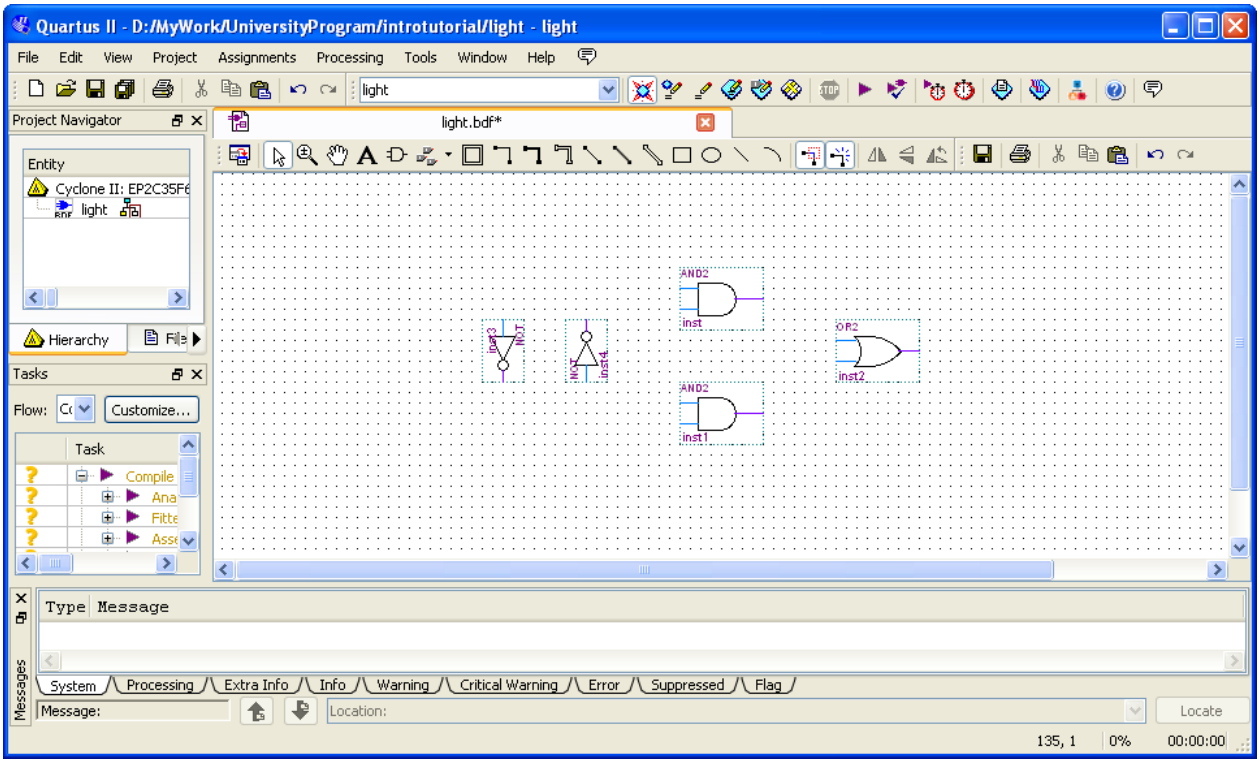


Figure 16. Import the gate symbols into the Graphic Editor window.

5.2 Importing Input and Output Symbols

Having entered the logic-gate symbols, it is now necessary to enter the symbols that represent the input and output ports of the circuit. Use the same procedure as for importing the gates, but choose the port symbols from the library *primitives/pin*. Import two instances of the input port and one instance of the output port, to obtain the image in Figure 17.

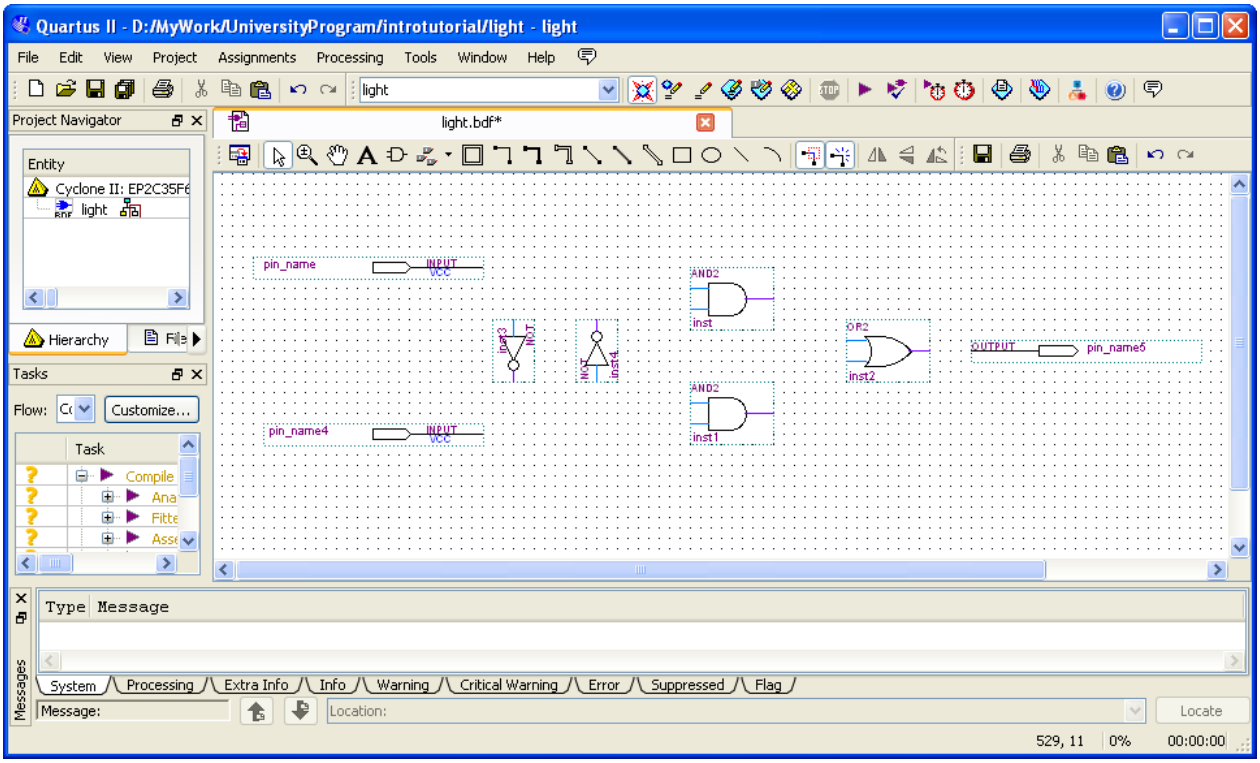


Figure 17. Import the input and output pins.

Assign names to the input and output symbols as follows. Make sure nothing is selected by clicking on an empty spot in the Graphic Editor window. Point to the top input symbol and double-click the mouse. The dialog box in Figure 18 will appear. Type the pin name, $x1$, and click OK. Similarly, assign the name $x2$ to the other input and f to the output. Alternatively, it is possible to change the name of an element by double-clicking on the name and typing a new one directly.

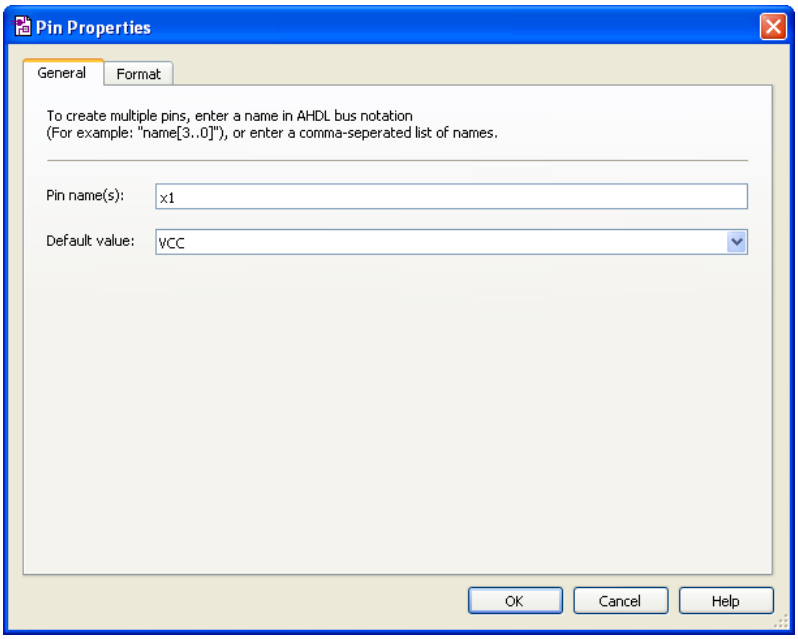


Figure 18. Naming of a pin.

5.3 Connecting Nodes with Wires

The symbols in the diagram have to be connected by drawing lines (wires). Click on the icon in the toolbar to activate the Orthogonal Node Tool. Position the mouse pointer over the right edge of the $x1$ input pin. Click and hold the mouse button and drag the mouse to the right until the drawn line reaches the pinstub on the top input of the AND gate. Release the mouse button when you see a box appear, which leaves the line connecting the two pinstubs. Next, draw a wire from the input pinstub of the leftmost NOT gate to touch the wire that was drawn above it. Note that a dot will appear indicating a connection between the two wires.

Use the same procedure to draw the remaining wires in the circuit. If a mistake is made, a wire can be selected by clicking on it, and removed by pressing the Delete key on the keyboard. Upon completing the diagram, click on the icon, to activate the Selection Tool. Now, changes in the appearance of the diagram can be made by selecting a particular symbol or wire and either moving it to a different location or deleting it. The final diagram is shown in Figure 19; save it.

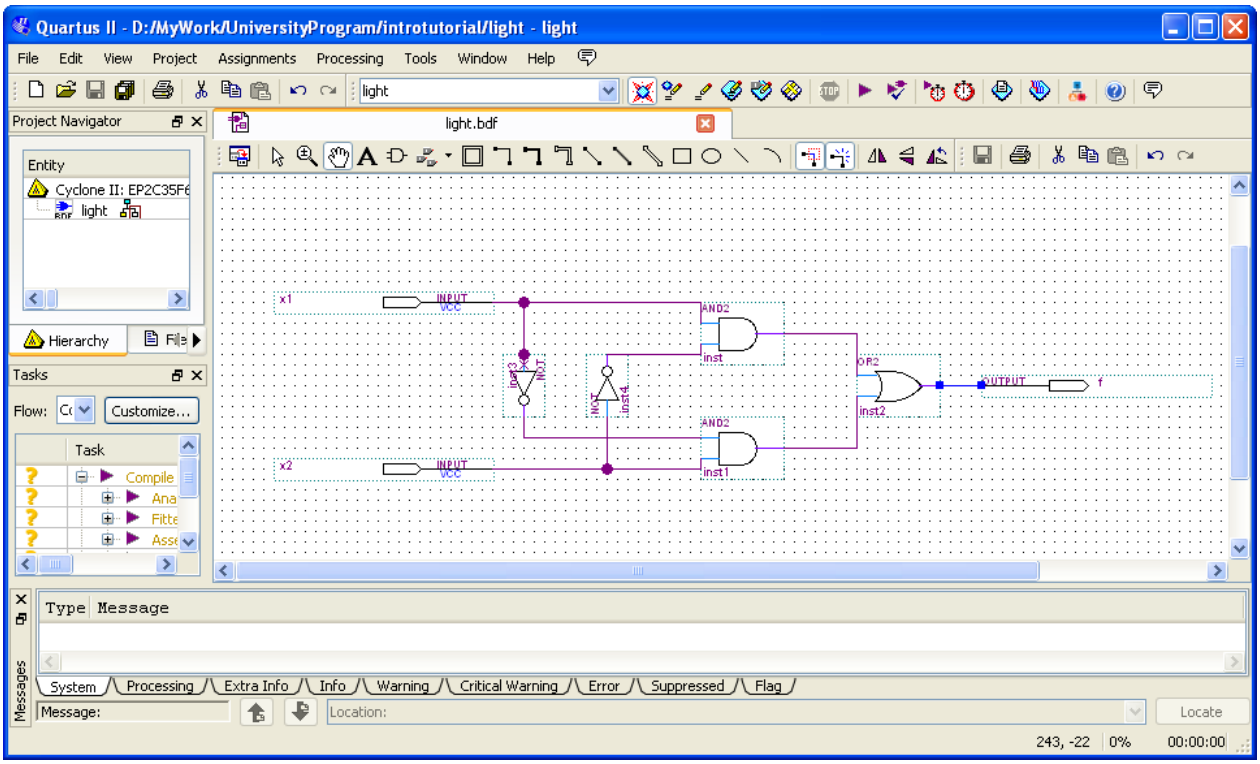


Figure 19. The completed schematic diagram.

6 Compiling the Designed Circuit

The entered schematic diagram file, *light.bdf*, is processed by several Quartus II tools that analyze the file, synthesize the circuit, and generate an implementation of it for the target chip. These tools are controlled by the application program called the *Compiler*.

Run the Compiler by selecting Processing > Start Compilation, or by clicking on the toolbar icon that looks like a purple triangle. Your project must be saved before compiling. As the compilation moves through various stages, its progress is reported in a window on the left side of the Quartus II display. Successful (or unsuccessful) compilation is indicated in a pop-up box. Acknowledge it by clicking OK, which leads to the Quartus II display in Figure 20. In the message window, at the bottom of the figure, various messages are displayed. In case of errors, there will be appropriate messages given.

When the compilation is finished, a compilation report is produced. A tab showing this report is opened automatically, as seen in Figure 20. The tab can be closed in the normal way, and it can be opened at any time either by selecting Processing > Compilation Report or by clicking on the icon . The report includes a number of sections listed on the left side. Figure 20 displays the Compiler Flow Summary section, which indicates that only one logic element and three pins are needed to implement this tiny circuit on the selected FPGA chip.

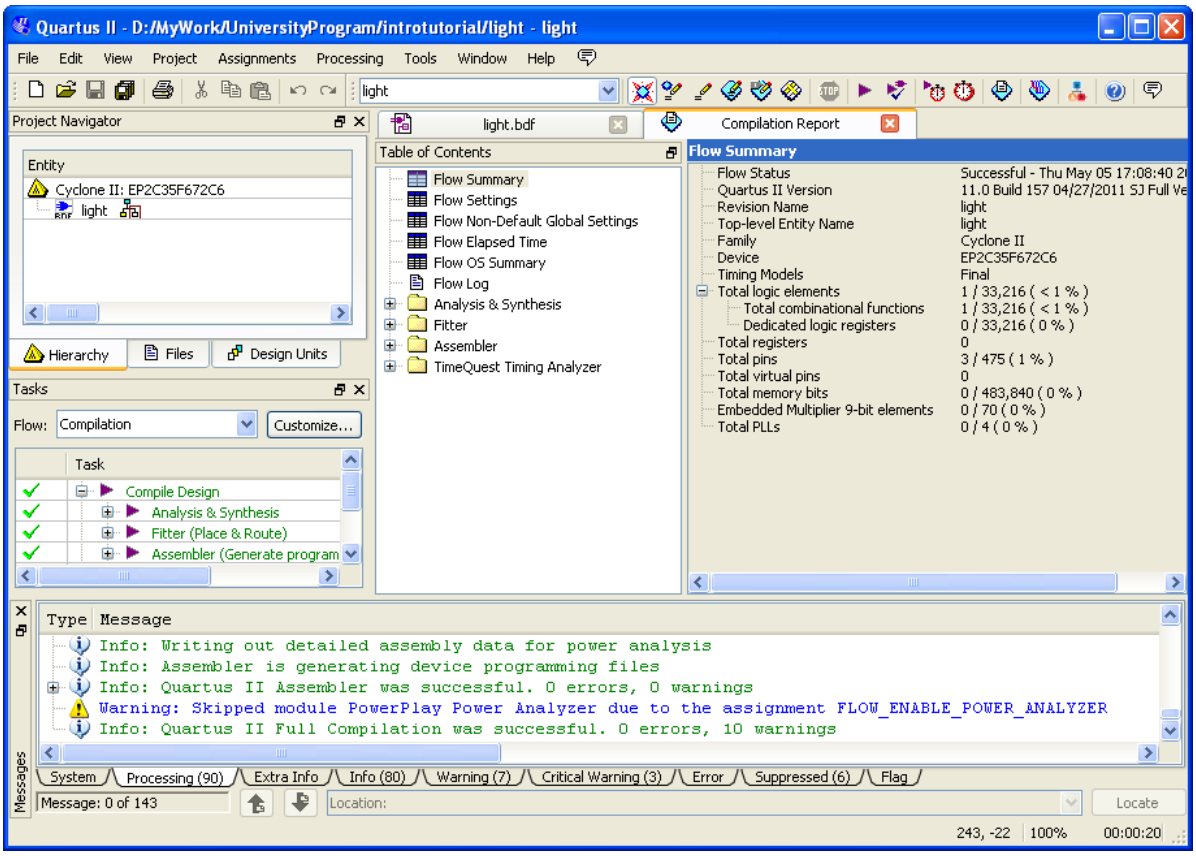

Quartus II - D:/MyWork/UniversityProgram/introtutorial/light - light
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Cyclone II: EP2C35F672C6
light
Hierarchy Files Design Units
Tasks
Flow: Compilation Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate program
light.bdf
Compilation Report
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Fitter
Assembler
TimeQuest Timing Analyzer
Flow Summary
Flow Status Successful - Thu May 05 17:08:40 2
Quartus II Version 11.0 Build 157 04/27/2011 SJ Full Ve
Revision Name light
Top-level Entity Name light
Family Cyclone II
Device EP2C35F672C6
Timing Models Final
Total logic elements 1 / 33,216 (< 1 %)
Total combinational functions 1 / 33,216 (< 1 %)
Dedicated logic registers 0 / 33,216 (0 %)
Total registers 0
Total pins 3 / 475 (1 %)
Total virtual pins 0
Total memory bits 0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements 0 / 70 (0 %)
Total PLLs 0 / 4 (0 %)
Type Message
Info: Writing out detailed assembly data for power analysis
Info: Assembler is generating device programming files
Info: Quartus II Assembler was successful. 0 errors, 0 warnings
Warning: Skipped module PowerPlay Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
Info: Quartus II Full Compilation was successful. 0 errors, 10 warnings
System Processing (90) Extra Info Info (80) Warning (7) Critical Warning (3) Error Suppressed (6) Flag
Message: 0 of 143
Location:
Locate
Messages
243, -22 100% 00:00:20

Figure 20. Display after a successful compilation.

6.1 Errors

Quartus II software displays messages produced during compilation in the Messages window. If the block diagram design file is correct, one of the messages will state that the compilation was successful and that there are no errors.

If the Compiler does not report zero errors, then there is at least one mistake in the schematic entry. In this case a message corresponding to each error found will be displayed in the Messages window. Double-clicking on an error message will highlight the offending part of the circuit in the Graphic Editor window. Similarly, the Compiler may display some warning messages. Their details can be explored in the same way as in the case of error messages. The user can obtain more information about a specific error or warning message by selecting the message and pressing the F1 function key.

To see the effect of an error, open the file *light.bdf*. Remove the wire connecting the output of the top AND gate to the OR gate. To do this, click on the icon, click the mouse on the wire to be removed (to select it) and press Delete. Compile the erroneous design by clicking on the icon. A pop-up box will ask if the changes made to the *light.bdf* file should be saved; click Yes. After trying to compile the circuit, Quartus II software will display a pop-up box indicating that the compilation was not successful. Acknowledge it by clicking OK. The compilation report

summary, given in Figure 21, now confirms the failed result. In the Table of Contents panel, expand the Analysis & Synthesis part of the report and then select Messages to have the messages displayed as shown in Figure 22. The Compilation Report can be dispayed as a separate window as in Figure 22 by right-clicking its tab and selecting Detach Window, and can be reattached by clicking Window > Attatch Window. Double-click on the first error message, which states that one of the nodes is missing a source. Quartus II software responds by displaying the *light.bdf* schematic and highlighting the OR gate which is affected by the error, as shown in Figure 23. Correct the error and recompile the design.

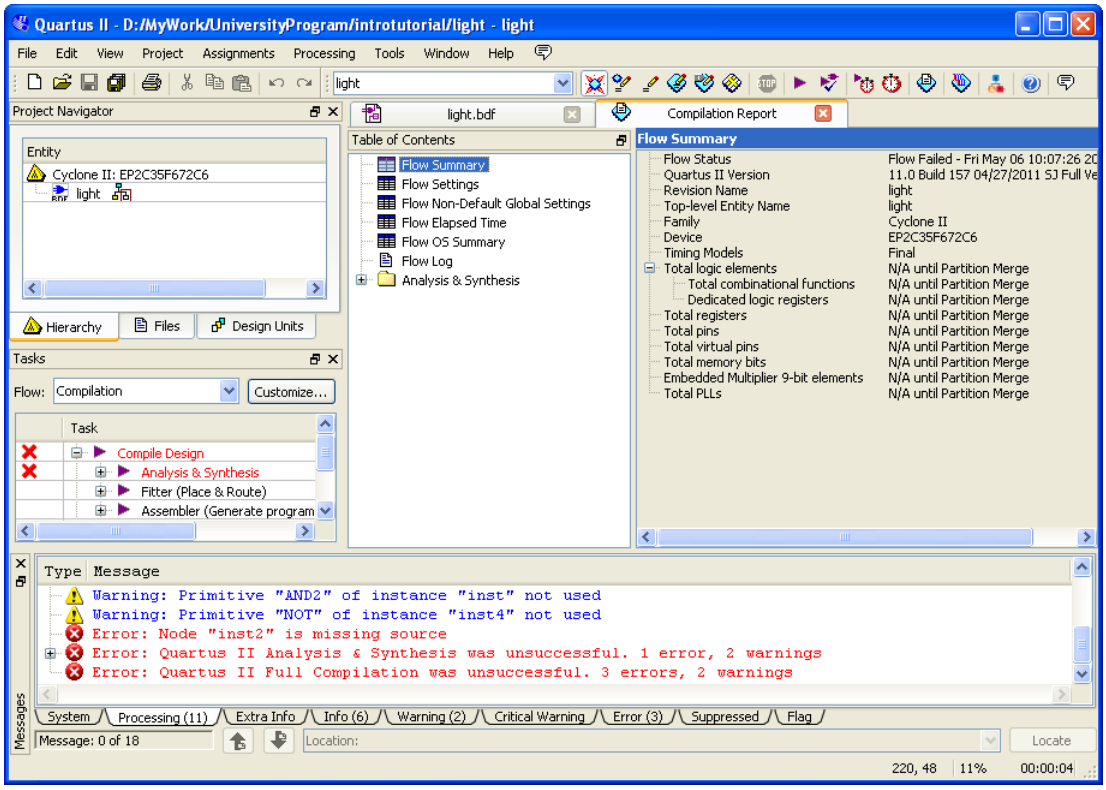

Quartus II - D:/MyWork/UniversityProgram/introtutorial/light - light
File Edit View Project Assignments Processing Tools Window Help
Project Navigator
Entity
Cyclone II: EP2C35F672C6
light
Hierarchy Files Design Units
Tasks
Flow: Compilation Customize...
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate program
light.bdf
Compilation Report
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Flow Summary
Flow Status Flow Failed - Fri May 06 10:07:26 20
Quartus II Version 11.0 Build 157 04/27/2011 SJ Full Ve
Revision Name light
Top-level Entity Name light
Family Cyclone II
Device EP2C35F672C6
Timing Models Final
Total logic elements N/A until Partition Merge
Total combinational functions N/A until Partition Merge
Dedicated logic registers N/A until Partition Merge
Total registers N/A until Partition Merge
Total pins N/A until Partition Merge
Total virtual pins N/A until Partition Merge
Total memory bits N/A until Partition Merge
Embedded Multiplier 9-bit elements N/A until Partition Merge
Total PLLs N/A until Partition Merge
Type Message
Warning: Primitive "AND2" of instance "inst" not used
Warning: Primitive "NOT" of instance "inst4" not used
Error: Node "inst2" is missing source
Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 2 warnings
Error: Quartus II Full Compilation was unsuccessful. 3 errors, 2 warnings
Messages
System Processing (11) Extra Info Info (6) Warning (2) Critical Warning Error (3) Suppressed Flag
Message: 0 of 18
Location:
Locate
220, 48 11% 00:00:04

Figure 21. Compilation report for the failed design.

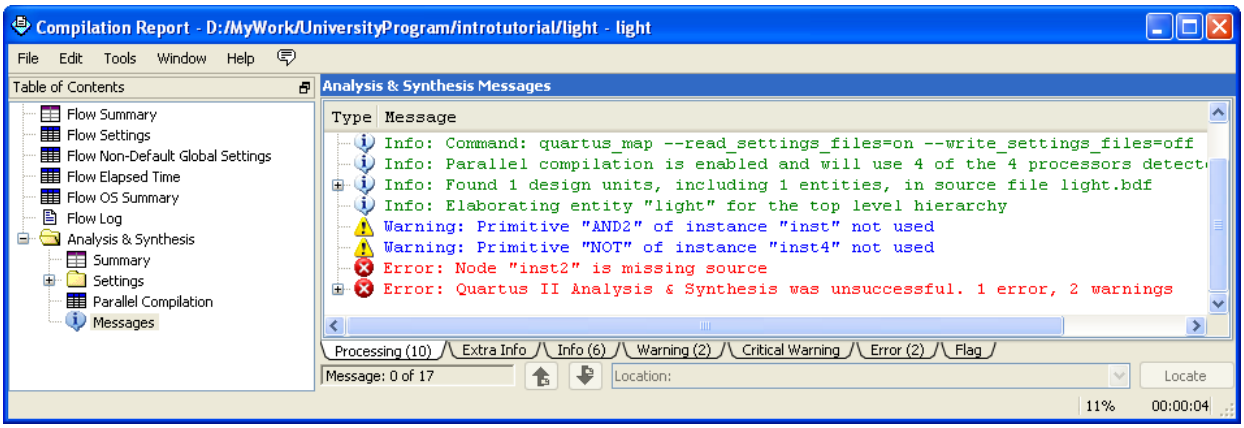

Compilation Report - D:/MyWork/UniversityProgram/introtutorial/light - light
File Edit Tools Window Help
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Summary
Settings
Parallel Compilation
Messages
Analysis & Synthesis Messages
Type Message
Info: Command: quartus_map --read_settings_files=on --write_settings_files=off
Info: Parallel compilation is enabled and will use 4 of the 4 processors detect
Info: Found 1 design units, including 1 entities, in source file light.bdf
Info: Elaborating entity "light" for the top level hierarchy
Warning: Primitive "AND2" of instance "inst" not used
Warning: Primitive "NOT" of instance "inst4" not used
Error: Node "inst2" is missing source
Error: Quartus II Analysis & Synthesis was unsuccessful. 1 error, 2 warnings
Processing (10) Extra Info Info (6) Warning (2) Critical Warning Error (2) Flag
Message: 0 of 17
Location:
Locate
11%
00:00:04

Figure 22. Error messages.

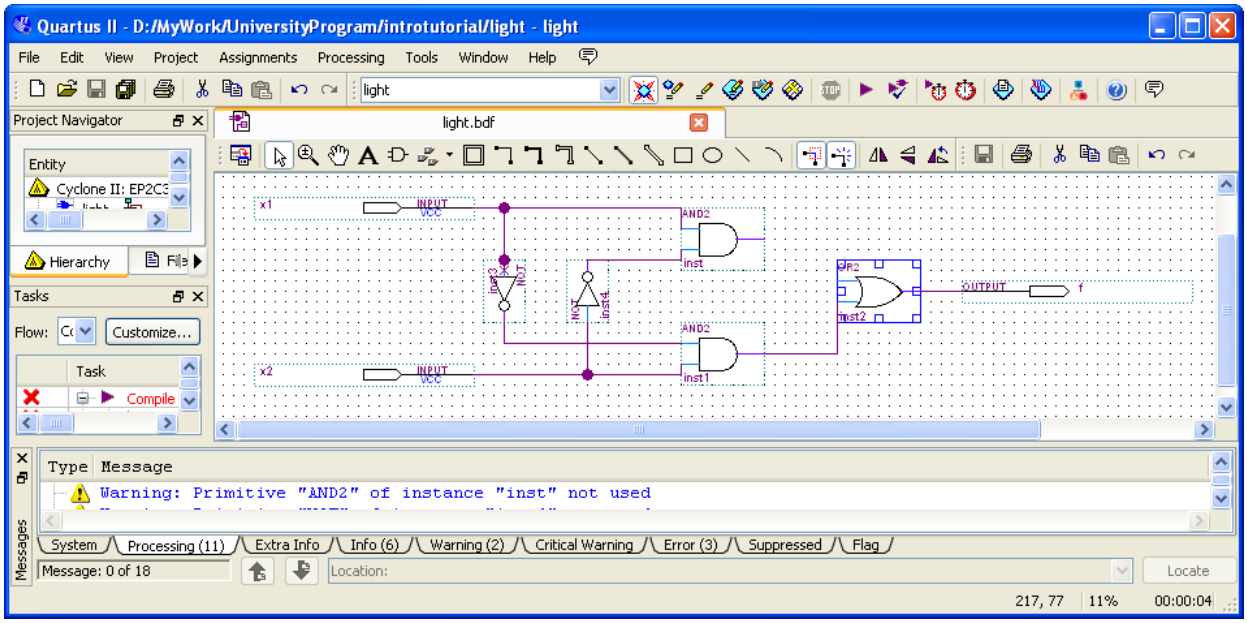


Figure 23. Identifying the location of the error.

7 Pin Assignment

During the compilation above, the Quartus II Compiler was free to choose any pins on the selected FPGA to serve as inputs and outputs. However, the DE-series board has hardwired connections between the FPGA pins and the other components on the board. We will use two toggle switches, labeled SW_0 and SW_1, to provide the external inputs, x_1 and x_2, to our example circuit. These switches are connected to the FPGA pins listed in Table 2. We will connect the output f to the green light-emitting diode labeled $LEDG_0$. Its FPGA pin assignment can also be found in Table

2.

Component	DE0	DE1	DE2	DE2-70	DE2-115
SW_0	PIN_J6	PIN_L22	PIN_N25	PIN_AA23	PIN_AB28
SW_1	PIN_H5	PIN_L21	PIN_N26	PIN_AB26	PIN_AC28
$LEDG_0$	PIN_J1	PIN_U22	PIN_AE22	PIN_W27	PIN_E21

Table 2. DE-Series Pin Assignments

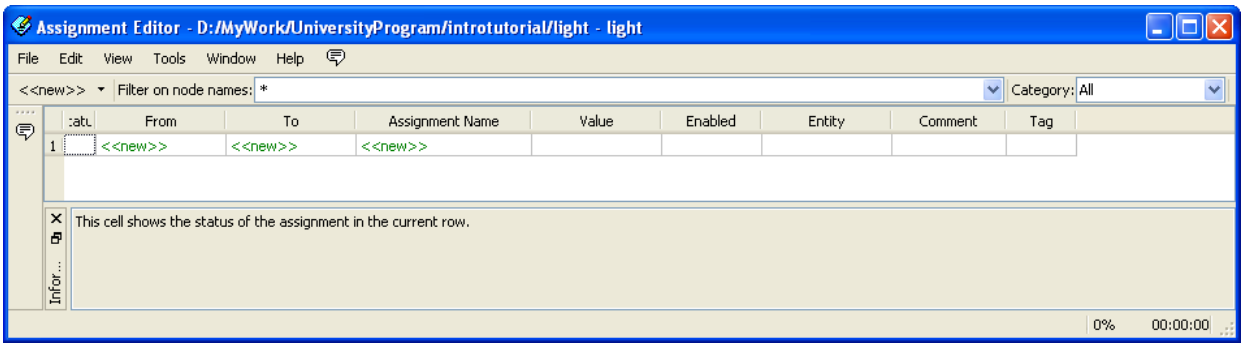

Assignment Editor - D:/MyWork/UniversityProgram/introtutorial/light - light
File Edit View Tools Window Help
<<new>> Filter on node names: *
Category: All
From To Assignment Name Value Enabled Entity Comment Tag
1 <<new>> <<new>> <<new>>
This cell shows the status of the assignment in the current row.
0% 00:00:00

Figure 24. The Assignment Editor window.

Pin assignments are made by using the Assignment Editor. Select Assignments > Assignment Editor to reach the window in Figure 24 (shown here as a detached window). In the Category drop-down menu select All. Click on the <<new>> button located near the top left corner to make a new item appear in the table. Double click the box under the column labeled To so that the Node Finder button appears. Click on the button (not the drop down arrow) to reach the window in Figure 25. In the Filter drop-down menu select Pins: all. Then click the List button to display the input and output pins to be assigned: f, $x1$, and $x2$. Click on $x1$ as the first pin to be assigned and click the > button; this will enter $x1$ in the Selected Nodes box. Click OK. $x1$ will now appear in the box under the column labeled To. Alternatively, the node name can be entered directly by double-clicking the box under the To column and typing in the node name.

Follow this by double-clicking on the box to the right of this new $x1$ entry, in the column labeled Assignment Name. Now, the drop-down menu in Figure 26 appears. Scroll down and select Location (Accepts wildcards/groups). Instead of scrolling down the menu to find the desired item, you can just type the first letter of the item in the Assignment Name box. In this case the desired item happens to be the first item beginning with L. Finally, double-click the box in the column labeled Value. Type the pin assignment corresponding to SW_0 for your DE-series board, as listed in Table 2.

Use the same procedure to assign input $x2$ and output f to the appropriate pins listed in Table 2. An example using a DE2 board is shown in Figure 27. To save the assignments made, choose File > Save. You can also simply close the Assignment Editor window, in which case a pop-up box will ask if you want to save the changes to assignments; click Yes. Recompile the circuit, so that it will be compiled with the correct pin assignments.

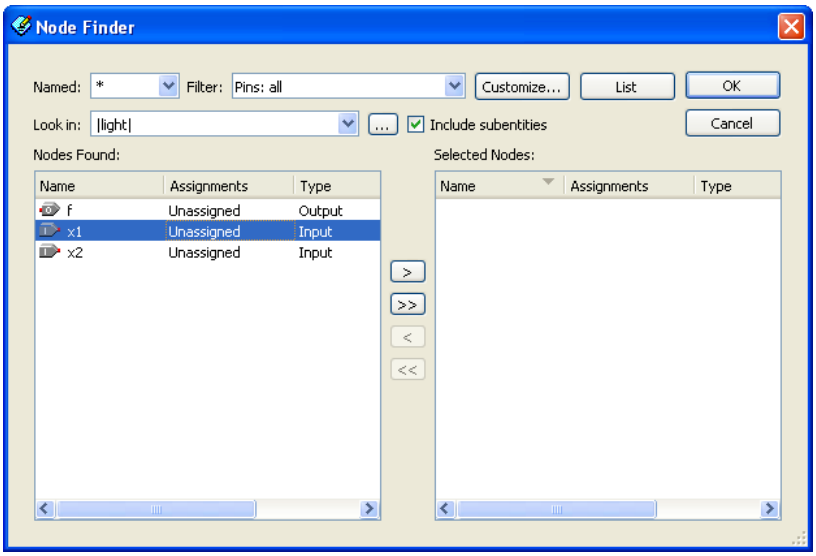


Figure 25. The Node Finder displays the input and output names.

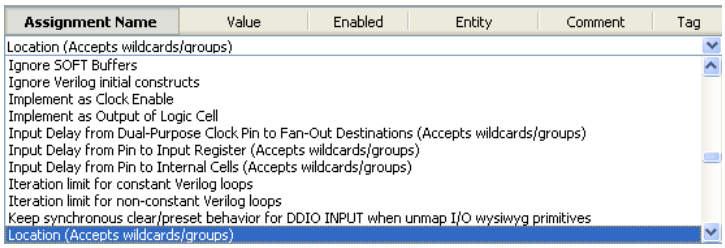

Assignment Name
Value
Enabled
Entity
Comment
Tag
Location (Accepts wildcards/groups)
Ignore SOFT Buffers
Ignore Verilog initial constructs
Implement as Clock Enable
Implement as Output of Logic Cell
Input Delay from Dual-Purpose Clock Pin to Fan-Out Destinations (Accepts wildcards/groups)
Input Delay from Pin to Input Register (Accepts wildcards/groups)
Input Delay from Pin to Internal Cells (Accepts wildcards/groups)
Iteration limit for constant Verilog loops
Iteration limit for non-constant Verilog loops
Keep synchronous clear/preset behavior for DDIO INPUT when unmap I/O wysiwyg primitives
Location (Accepts wildcards/groups)

Figure 26. The available assignment names for a DE-series board.

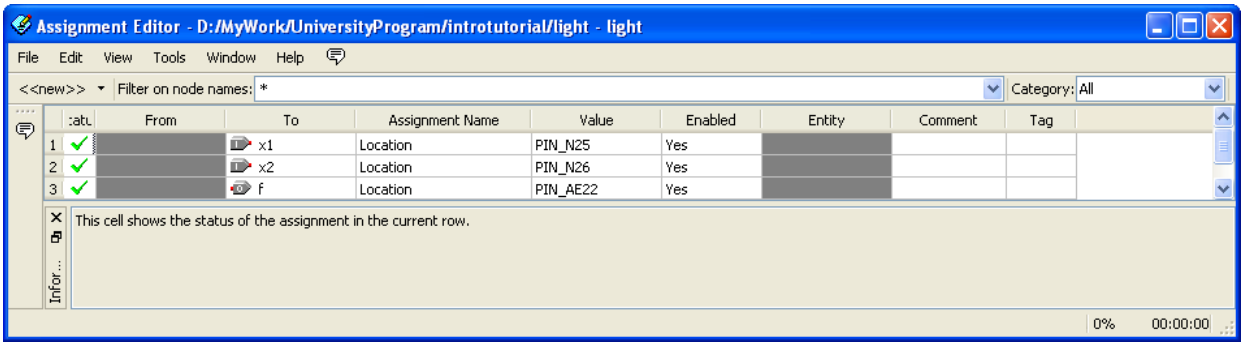


Figure 27. The complete assignment.

The DE-series board has fixed pin assignments. Having finished one design, the user will want to use the same pin assignment for subsequent designs. Going through the procedure described above becomes tedious if there are many pins used in the design. A useful Quartus II feature allows the user to both export and import the pin assignments from a special file format, rather than creating them manually using the Assignment Editor. A simple file format that can be used for this purpose is the *Quartus II Settings File (QSF)* format. The format for the file for our simple project (on a DE2 board) is

```
set_location_assignment PIN_N25 -to x1
set_location_assignment PIN_N26 -to x2
set_location_assignment PIN_AE22 -to f
```

By adding lines to the file, any number of pin assignments can be created. Such *qsf* files can be imported into any design project.

If you created a pin assignment for a particular project, you can export it for use in a different project. To see how this is done, open again the Assignment Editor to reach the window in Figure 27. Select Assignments > Export Assignment which leads to the window in Figure 28. Here, the file *light.qsf* is available for export. Click on OK. If you now look in the directory, you will see that the file *light.qsf* has been created.

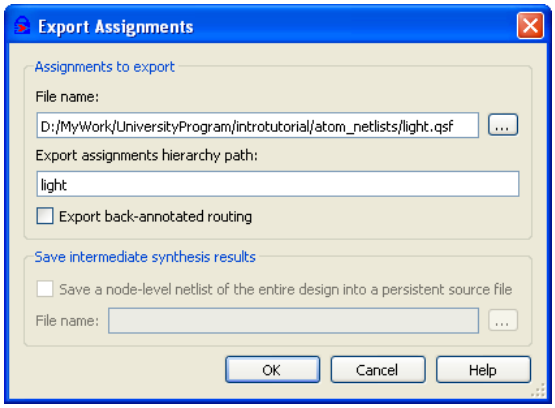


Figure 28. Exporting the pin assignment.

You can import a pin assignment by choosing Assignments > Import Assignments. This opens the dialogue in Figure 29 to select the file to import. Type the name of the file, including the *qsf* extension and the full path to the directory that holds the file, in the File Name box and press OK. Of course, you can also browse to find the desired file.

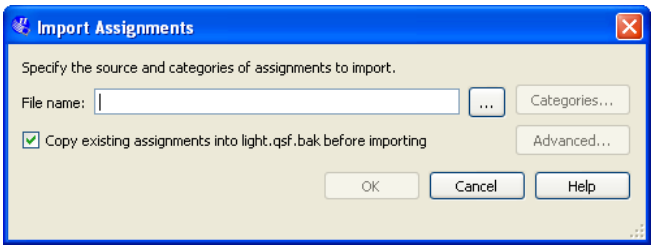


Figure 29. Importing the pin assignment.

For convenience when using large designs, all relevant pin assignments for the DE-series board are given in individual files. For example, the DE2 pin assignments can be found in the *DE2_pin_assignments.qsf* file, in the directory *tutorials\design_files*, which is included on the CD-ROM that accompanies the DE-series board and can also be found on Altera's DE-series web pages. This file uses the names found in the *DE2 User Manual*. If we wanted to make the pin assignments for our example circuit by importing this file, then we would have to use the same names in our Block Diagram/Schematic design file; namely, *SW[0]*, *SW[1]* and *LEDG[0]* for *x1*, *x2* and *f*, respectively. Since these signals are specified in the *DE2_pin_assignments.qsf* file as elements of vectors *SW* and *LEDG*, we must refer to them in the same way in our design file. For example, in the *DE2_pin_assignments.qsf* file the 18 toggle switches are called *SW[17]* to *SW[0]*. In a design file they can also be referred to as a vector *SW[17..0]*.

8 Simulating the Designed Circuit

Before implementing the designed circuit in the FPGA chip on the DE-series board, it is prudent to simulate it to ascertain its correctness. The QSim tools can be used to simulate the behavior of a designed circuit. Before the circuit can be simulated, it is necessary to create the desired waveforms, called *test vectors*, to represent the input signals. It is also necessary to specify which outputs, as well as possible internal points in the circuit, the designer wishes to observe. The simulator applies the test vectors to a model of the implemented circuit and determines the expected response. We will use the Quartus II Waveform Editor to draw the test vectors, as follows:

Edit for IS1500: You cannot start QSim from the Start Menu. See the Laboratory Exercise Instructions (lab-PM).

1. ~~Select Start > All Programs > Altera > University Program > Simulation Tools > QSim~~ to open the QSim tools, which will display the window in Figure 30.

2. Select File > Open Project to display a popup window in which you can browse your directories and choose a project file (*.qpf* file). Select the project you wish to simulate and clickOK.

3. Generate the node finder files by selecting Processing > Generate Node Finder Files

4. From QSim, open the Waveform Editor window by selecting File > New Simulation Input File.

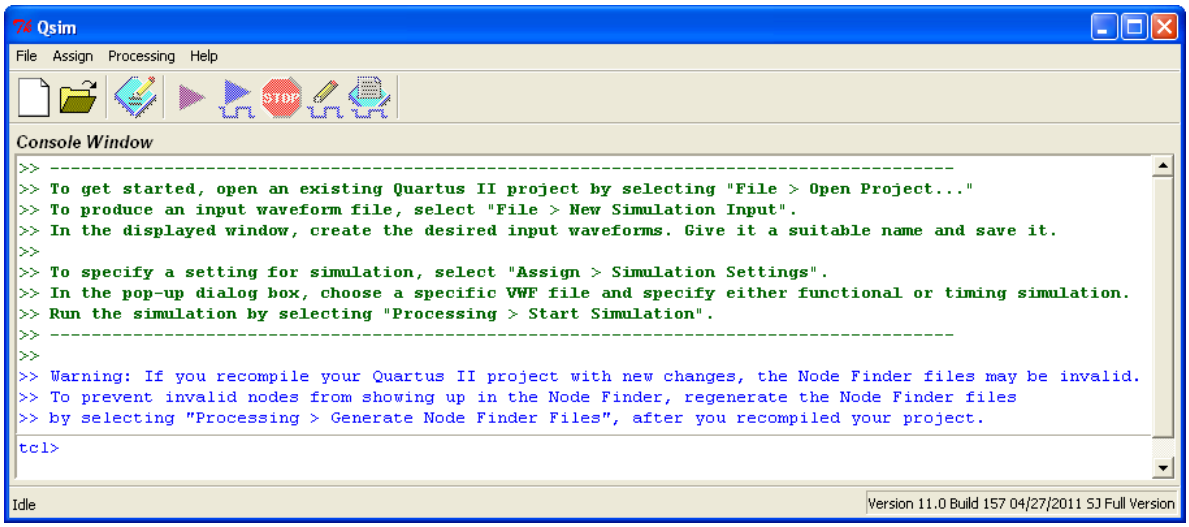

Qsim
File Assign Processing Help
Console Window
>> ----------
>> To get started, open an existing Quartus II project by selecting "File > Open Project..."
>> To produce an input waveform file, select "File > New Simulation Input".
>> In the displayed window, create the desired input waveforms. Give it a suitable name and save it.
>>
>> To specify a setting for simulation, select "Assign > Simulation Settings".
>> In the pop-up dialog box, choose a specific VWF file and specify either functional or timing simulation.
>> Run the simulation by selecting "Processing > Start Simulation".
>> ----------
>>
>> Warning: If you recompile your Quartus II project with new changes, the Node Finder files may be invalid.
>> To prevent invalid nodes from showing up in the Node Finder, regenerate the Node Finder files
>> by selecting "Processing > Generate Node Finder Files", after you recompiled your project.
tcl>
Idle
Version 11.0 Build 157 04/27/2011 SJ Full Version

Figure 30. The QSim Window.

5. The Waveform Editor window is depicted in Figure 31. Save the file under the name *light.vwf*; note that this changes the name in the displayed window. Set the desired simulation to run from 0 to 200 ns by selecting Edit > Set End Time and entering 200 ns in the dialog box that pops up. Selecting View > Fit in Window displays the entire simulation range of 0 to 200 ns in the window, as shown in Figure 32. You may wish to resize the window to its maximum size.

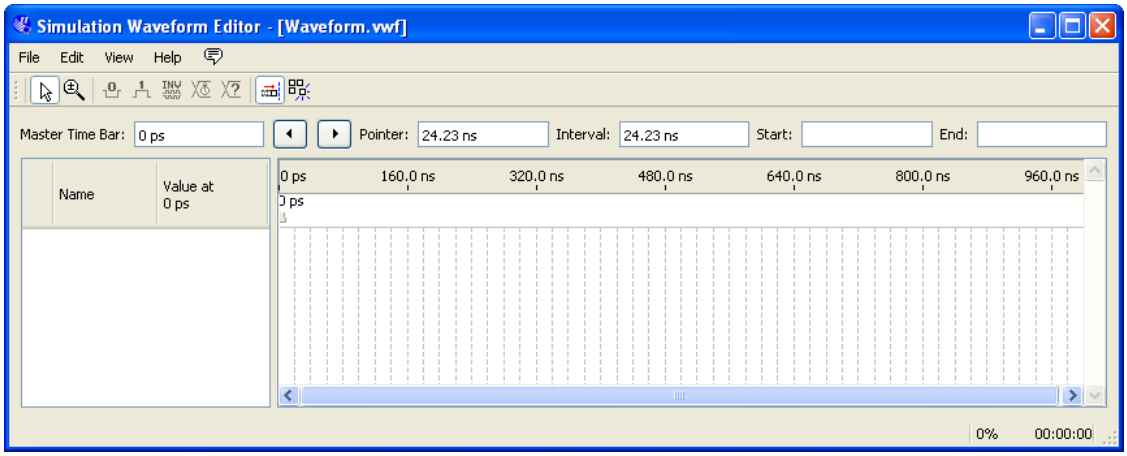


Figure 31. The Waveform Editor window.

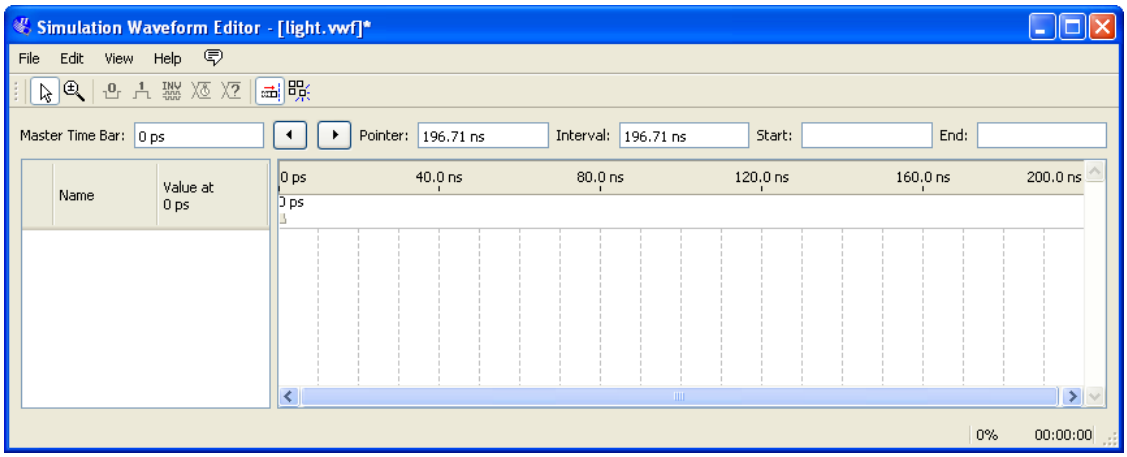


Figure 32. The augmented Waveform Editor window.

6. Next, we want to include the input and output nodes of the circuit to be simulated. Click Edit > Insert > Insert Node or Bus to open the window in Figure 33. It is possible to type the name of a signal (pin) into the Name box, or use the Node Finder to search your project for the signals. Click on the button labeled Node Finder to open the window in Figure 34. The Node Finder utility has a filter used to indicate what type of nodes are to be found. Since we are interested in input and output pins, set the filter to Pins: all. Click the List button to find the input and output nodes as indicated on the left side of the figure.

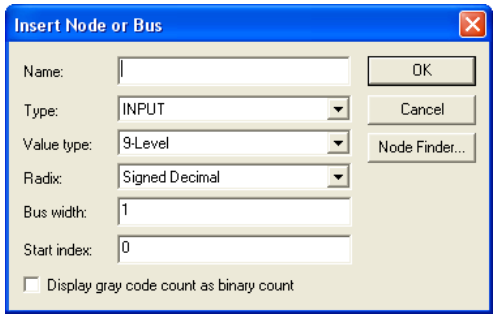


Figure 33. The Insert Node or Bus dialogue.

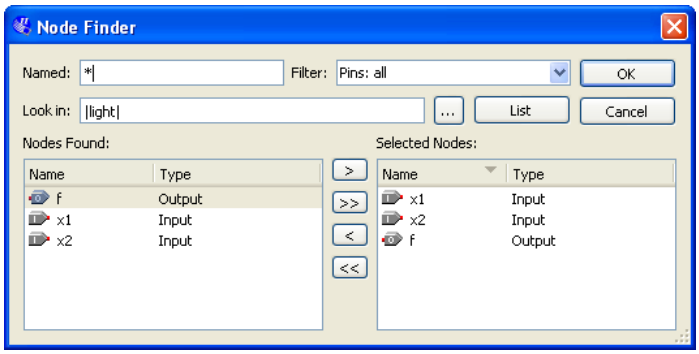


Figure 34. Selecting nodes to insert into the Waveform Editor.

Click on the *x1* signal in the Nodes Found box in Figure 34, and then click the > sign to add it to the Selected Nodes box on the right side of the figure. Do the same for *x2* and *f*. Click OK to close the Node Finder window, and then click OK in the window of Figure 33. This leaves a fully displayed Waveform Editor window, as shown in Figure 35.

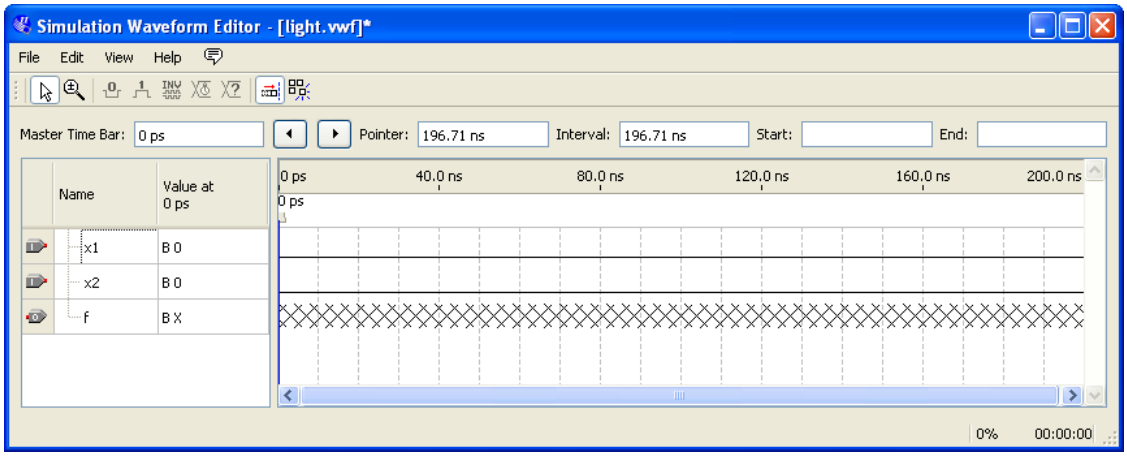


Figure 35. The nodes needed for simulation.

7. We will now specify the logic values to be used for the input signals *x1* and *x2* during simulation. The logic values at the output *f* will be generated automatically by the simulator. To make it easy to draw the desired waveforms, the Waveform Editor displays (by default) vertical guidelines and provides a drawing feature that snaps on these lines (which can otherwise be invoked by choosing the Snap To Grid button). Observe also a solid vertical line, which can be moved by pointing to its top and dragging it horizontally. This reference line is used in analyzing the timing of a circuit; move it to the *time* = 0 position. The waveforms can be drawn using the Selection Tool, which is activated by selecting the icon in the toolbar.

   To simulate the behavior of a large circuit, it is necessary to apply a sufficient number of input valuations and observe the expected values of the outputs. In a large circuit the number of possible input valuations

may be huge, so in practice we choose a relatively small (but representative) sample of these input valuations. However, for our tiny circuit we can simulate all four input valuations given in Figure 11. We will use four 50-ns time intervals to apply the four test vectors.

We can generate the desired input waveforms as follows. Click on the waveform for the *x1* node. Once a waveform is selected, the editing commands in the Waveform Editor can be used to draw the desired waveforms. Commands are available for setting a selected signal to 0, 1, arbitrary value, inverting its existing value (INV), or defining a clock waveform. Each command can be activated by using the Edit > Value command, or via the toolbar for the Waveform Editor. The Edit menu can also be opened by right-clicking on a waveform.

Set *x1* to 0 in the time interval 0 to 100 ns, which is probably already set by default. Next, set *x1* to 1 in the time interval 100 to 200 ns. Do this by pressing the mouse at the start of the interval and dragging it to its end, which highlights the selected interval, and choosing the logic value 1 in the toolbar. Make *x2* = 1 from 50 to 100 ns and also from 150 to 200 ns, which corresponds to the truth table in Figure 11. This should produce the image in Figure 36. Observe that the output *f* is displayed as having an unknown value at this time, which is indicated by a hashed pattern; its value will be determined during simulation. Save the file.

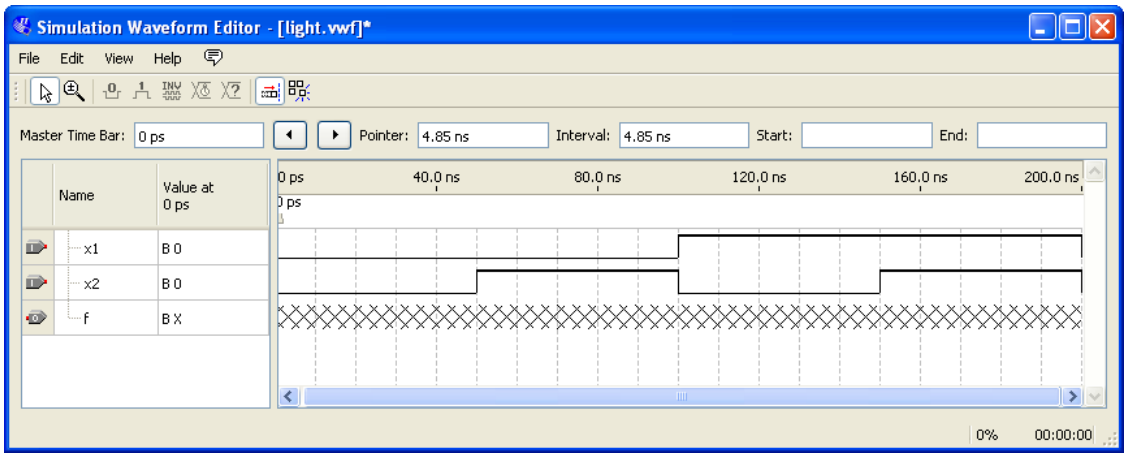


Figure 36. Setting of test values.

8.1 Performing the Simulation

A designed circuit can be simulated in two ways. The simplest way is to assume that logic elements and interconnection wires in the FPGA are perfect, thus causing no delay in propagation of signals through the circuit. This is called *functional simulation*. A more complex alternative is to take all propagation delays into account, which leads to *timing simulation*. Typically, functional simulation is used to verify the functional correctness of a circuit as it is being designed. This takes much less time, because the simulation can be performed simply by using the logic expressions that define the circuit.

8.1.1 Functional Simulation

To perform the functional simulation, return to the QSim Window and select Assign > Simulation Settings... to open the Simulation Settings window in Figure 37. Click the Browse button and select the *light.vwf* file you created.

Choose Functional as the simulation type, and click OK. Before running the functional simulation it is necessary to create the required netlist, which is done by selecting Processing > Generate Functional Simulation Netlist. A simulation run is started by Processing > Start Simulation, or by using the icon . At the end of the simulation, Quartus II software indicates its successful completion and displays a Simulation Report illustrated in Figure 38. If your report window does not show the entire simulation time range, click on the report window to select it and choose View > Fit in Window. Observe that the output f is as specified in the truth table of Figure 11.

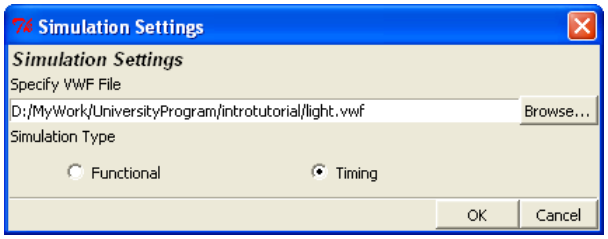


Figure 37. Specifying the simulation type.

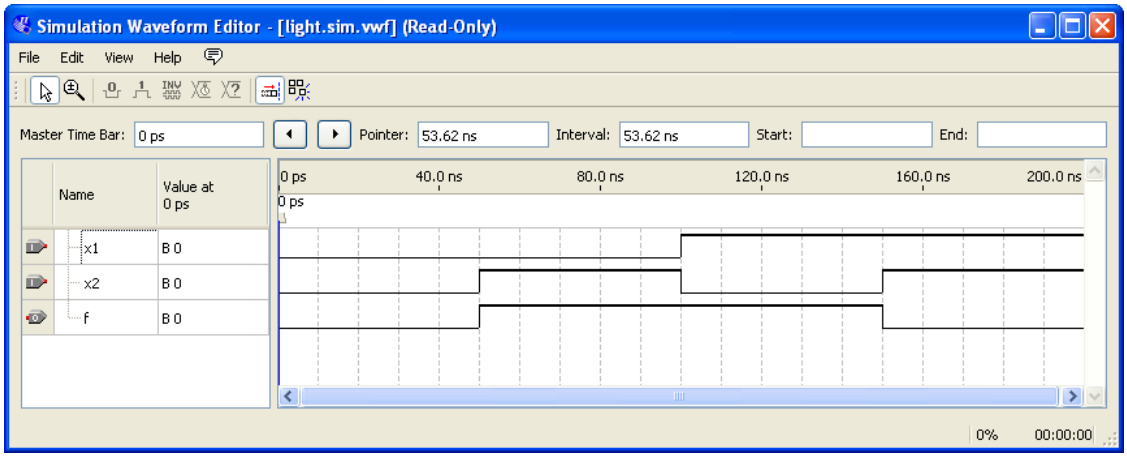


Figure 38. The result of functional simulation.

8.1.2 Timing Simulation

Having ascertained that the designed circuit is functionally correct, we should now perform the timing simulation to see how it will behave when it is actually implemented in the chosen FPGA device. Select Assign > Simulation Settings... to get to the window in Figure 37, choose Timing as the simulation type, and click OK. Run the simulator, which should produce the waveforms in Figure 39. Observe that there is a delay of about 6 ns in producing a change in the signal f from the time when the input signals, x_1 and x_2, change their values. This delay is due to the propagation delays in the logic element and the wires in the FPGA device.

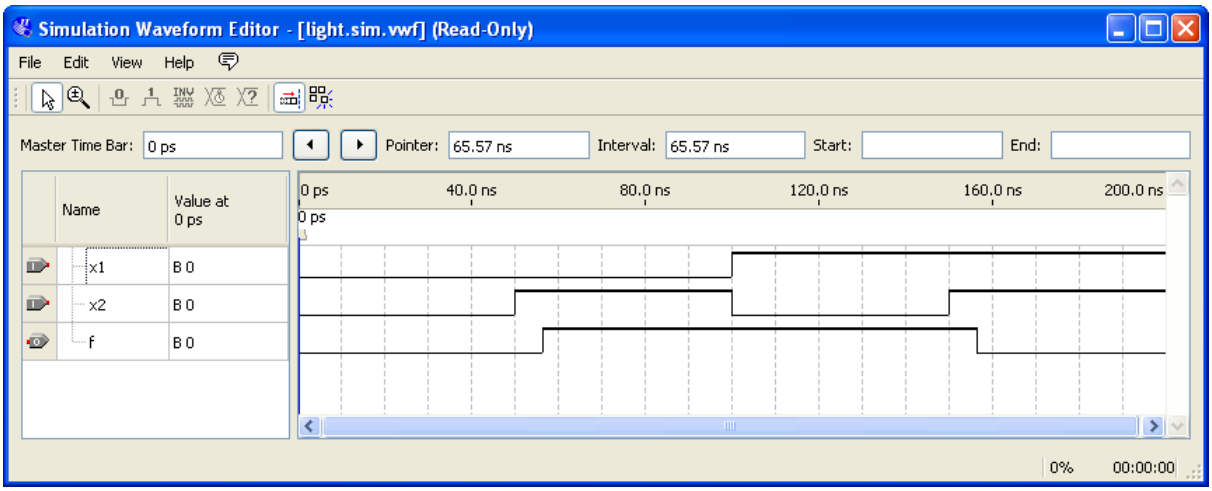


Figure 39. The result of timing simulation.

9 Programming and Configuring the FPGA Device

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus II Compiler's Assembler module. Altera's DE-series board allows the configuration to be done in two different ways, known as JTAG and AS modes. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board. To use this connection, it is necessary to have the USB-Blaster driver installed. If this driver is not already installed, consult the tutorial *Getting Started with Altera's DE-Series Boards* for information about installing the driver. Before using the board, make sure that the USB cable is properly connected and turn on the power supply switch on the board.

In the JTAG mode, the configuration data is loaded directly into the FPGA device. The acronym JTAG stands for Joint Test Action Group. This group defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. If the FPGA is configured in this manner, it will retain its configuration as long as the power remains turned on. The configuration information is lost when the power is turned off. The second possibility is to use the Active Serial (AS) mode. In this case, a configuration device that includes some flash memory is used to store the configuration data. Quartus II software places the configuration data into the configuration device on the DE-series board. Then, this data is loaded into the FPGA upon power-up or reconfiguration. Thus, the FPGA need not be configured by the Quartus II software if the power is turned off and on. The choice between the two modes is made by the RUN/PROG switch on the DE-series board. The RUN position selects the JTAG mode, while the PROG position selects the AS mode.

9.1 JTAG Programming

The programming and configuration task is performed as follows. Flip the RUN/PROG switch into the RUN position. Select Tools > Programmer to reach the window in Figure 40. Here it is necessary to specify the programming

hardware and the mode that should be used. If not already chosen by default, select JTAG in the Mode box. Also, if the USB-Blaster is not chosen by default, press the Hardware Setup... button and select the USB-Blaster in the window that pops up, as shown in Figure 41.

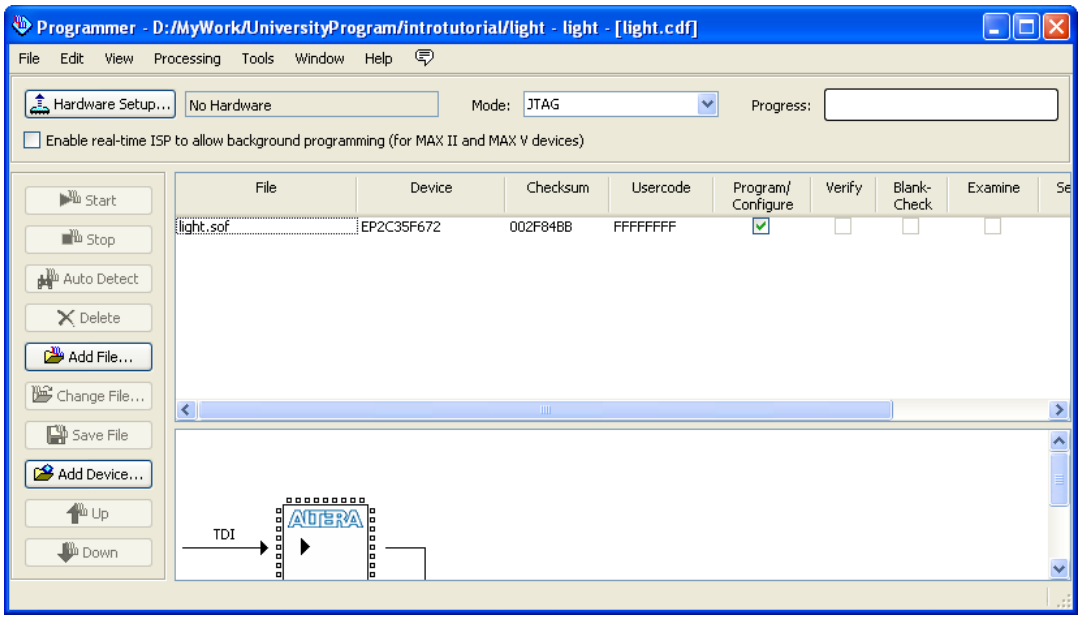


Figure 40. The Programmer window.

Observe that the configuration file *light.sof* is listed in the window in Figure 40. If the file is not already listed, then click Add File and select it. This is a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device. The extension *.sof* stands for SRAM Object File. Click on the Program/Configure check box, as shown in Figure 42.

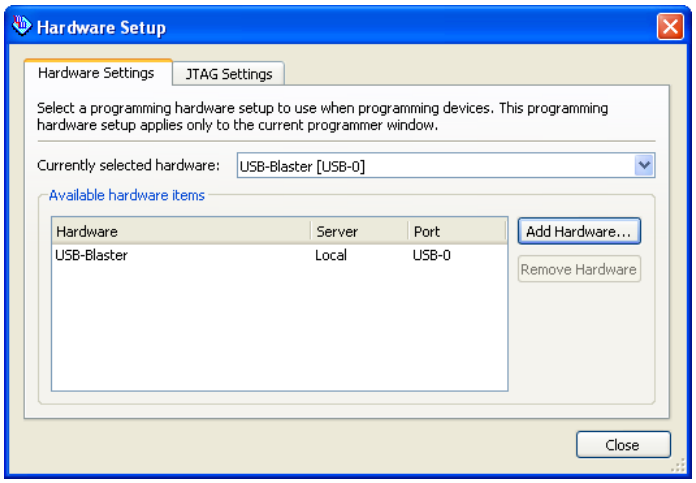


Figure 41. The Hardware Setup window.

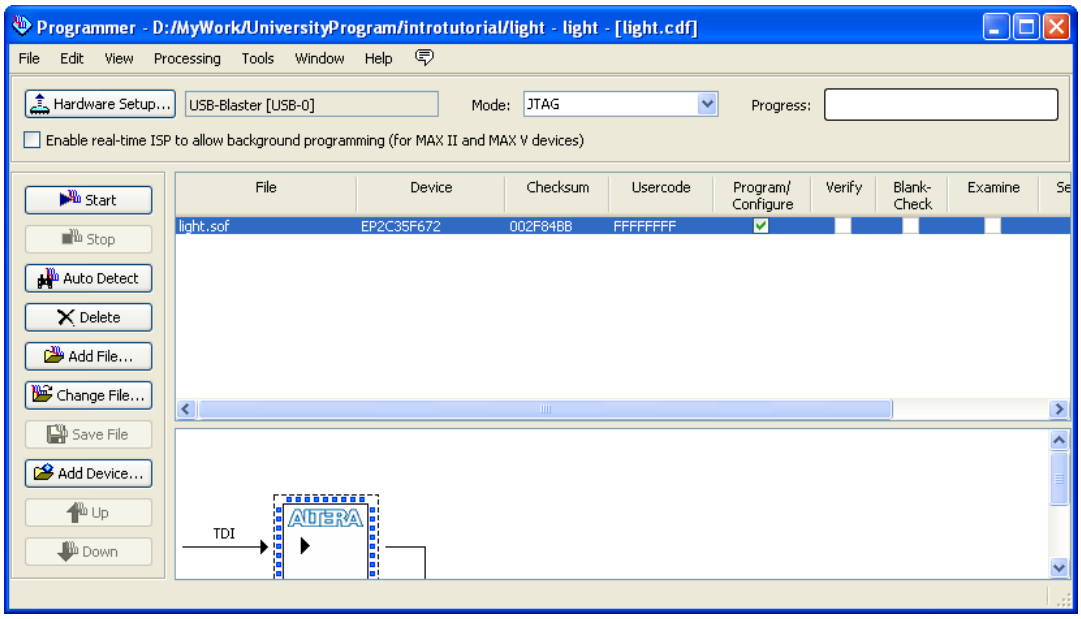


Figure 42. The updated Programmer window.

Now, press Start in the window in Figure 42. An LED on the board will light up when the configuration data has been downloaded successfully. If you see an error reported by Quartus II software indicating that programming failed, then check to ensure that the board is properly powered on.

Edit for IS1500 at KTH: pages 32-36 deleted.

Copyright ©1991-2011 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

This document is being provided on an "as-is" basis and as an accommodation and therefore all warranties, representations or guarantees of any kind (whether express, implied or statutory) including, without limitation, warranties of merchantability, non-infringement, or fitness for a particular purpose, are specifically disclaimed.